Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Environmental Sciences 8 (2011) 319 – 327

Procedia Environmental Sciences

ICESB 2011: 25-26 November 2011, Maldives

A Collaboration Transparence Approach to Share Heterogeneous Single-User Molecule Editors

Chen Zhao[a*], Ruisheng Zhang[b], Ramin Yahyapour[c], Rongjing Hu[b]

[a]*School of Mathematics and Statistics,Lanzhou University, Lanzhou 730000, China*
[b]*School of Information Science & Engineering ,Lanzhou University Lanzhou 730000, China*
[c]*IT & Media Center, Technical University Dortmund, Dortmund 44227, Germany*

Abstract

In the biology and chemistry field various types of molecule editors exist. It is convenient for biologist and chemist to use those editors to create and modify representations of molecular structures. Most editors however are designed for use by a single user only. Thus biologists and chemists lack tools for collaborative work. In this paper we present a transparent approach for collaboration. This approach is used to share off-the-shelf single-user applications without modifying the source code and thus to provide those editors with groupware capabilities. On this basis we go a step further by also designing and implementing a mechanism to make biologists and chemists who use heterogeneous single-user molecule editors collaborate with each other with their favorite tools instead of using just one version. Our research can reuse an amount of excellent existing molecule editors. This makes it easy to collaborate for biologists and chemists with a familiar interface. Additionally it saves money and time because it is not necessary to develop new tools from scratch.

*Keywords:*molecule editor; collaboration transparence; heterogeneous; computational biology

1. Introduction

With the increasing importance of CSCW (Computer-supported cooperative work) in domain of Computational Biology, and the demand for processing large amounts biological and chemical data on distributed system, such as Grids or Clouds [1], etc, the group collaboration becomes an indispensable work mode for biologists and chemists. Accordingly the development of group editors [2] specific for accomplishing biological and chemical activities should be considered, since those are an important

* Chen Zhao. Tel.:+49-231-755-2346 fax: +49-231-755-2731.
E-mail address: ccc132@post.uni-dortmund.de

doi:10.1016/j.proenv.2011.10.051

category of interactive groupware applications that allow a distributed group of people to work together on a set of shared data objects (document) [3]. However, there are some differences in cooperative modes between the domain of biology, chemistry and the generic approach., the objects operated upon not only include text, graphics, which are common for general groupware (e.g. virtual whiteboards), but also include some complex 2D or 3D molecular structures, which represent the arrangement of the atoms that constitute a molecule. These determine several properties of a substance including its reactivity, polarity, phase of matter, color, magnetism, and biological activity [4][5]. As such existing collaboration applications cannot satisfy the needs of biologist and chemist for collaboration.

A biology and chemistry-oriented group-collaboration solution needs to be developed, which should support synchronous collaboration, Interactions in such an environment should take place synchronously since interactions of collaborators are simultaneous or separated only by short periods of time. An asynchronous communication approach would be better suited for operations that are separated by relatively long periods of time [6]. The collaborative environment should also carry out a series of biology and chemistry-oriented tasks, for example for creating and modifying representations of molecular structure.

There are two basic choices how to develop synchronous collaborative applications. Collaboration-aware applications are specifically designed for simultaneous use by multiple users. Collaboration-transparent applications instead provide existing single-user applications with collaboration capabilities [7] [8]. Collaboration-aware applications are rather focused on the collaborative management and organization. A weakness of such systems may be that the developers have brought up more attention to creating collaborative features while neglecting the applications core-domain features [9]. Single-user applications in contrast are more mature regarding their feature set and their stability. Transparence of collaboration allows users to use their existing single-user applications to collaborate on tasks for which no collaborative environments are available i.e. multiple users can share a single-user application. As such it is more promising to extend established productivity tools with collaborative features [6].

Still the approach of creating a collaborative environment from a single-user application bears challenges e.g. not every biologist and chemist is experienced with a particular application and all users have their own custom workflows and setups. Forcing all participants to use the same application may prove counterproductive and result in resistance from the user-pier. For example asking a group of office users who would like to abandon their favorite word processor for a collaborative tool might result in negative responses. [6][10].

Therefore it is important to extend existing key applications, which are widely used, with collaboration-enabling features. This path however introduces new technical challenges to collaborative systems. For example, if the system only shares one application, it usually ensures the same input to each client of the same application. In contrast, if the system shares multi-applications, in different client, equivalent semantic commands of different applications are usually implemented by different input sequences. A mechanism which implements semantic translation needs to be researched to find a solution for this problem. Maintaining consistency is an important issue in groupware systems [11]. If a single-user application is to be shared among a number of users problems regarding concurrent access to arbitrary object within the application needs to be resolved. Finally the organization of the interactions that finally form the collaboration needs to be defined and resolved as well.

2. Relative Work

There are many excellent single-user applications which can complete many tasks in the biology and chemistry domain. Creating two-dimensional representations of molecules and chemical reactions can be

created with those applications efficiently and conveniently. Those renderings can be used as illustrations and for inquiries in databases [12]. Three-dimensional molecule editors – usually part of molecule modeling software packages – are used to build molecular models. They are powerful and widely applied to various calculations processes. For example, biologist and chemists use some of them to construct protein models for applications in Virtual Screening [13] procedures. They are indispensable tools for modern biologists and chemists. Some of those applications that are widely used in the domain of computational biology are JChemPaint [14], ChemAxon [15], Leatherface [16].

There has been some research in processing heterogeneous interoperation. DistEdit[17] for example is a powerful toolkit developed for converting existing single-user editors to groupware. To apply it however it is necessary to modify the source code of the base-applications. Most single user editors however only provide an API instead of full source code. PSI (Platform for Shared Interaction) is an infrastructure to support the dynamic sharing of information across a range of cooperative environments [18]. It can process interactions between heterogeneous applications without modification requiring modifications to the applications source-code. PSI nevertheless is insufficient for implementing tightly coupled synchronous collaborations and does not satisfy the need of multi-format conversion for sharing molecule editor.

3. Challenges of Sharing Heterogeneous single-user molecule Editor Applications

Normally, building group systems requires solutions to solving problems in distributed concurrency control, group interfaces and group organization [19]. But there are several differences in sharing the existing heterogeneous molecule editors.

In general, there are many differences between molecule editors which were developed by different teams. These differences rank from knowledge representation to function implementation. For example, most molecule editors use special file formats to represent biological and chemical domain knowledge, which contains a number of kinds of information about a particular compound, from its physical atomic structure and elemental makeup to its water-solubility, melting point, atomic weight, and various spectral descriptions. Markup languages such as the Chemical Markup Language (CML) [20] and a set of proprietary file formats exist for this. Although there are some powerful editors that can read and edit multiple of those formats, there still are many discrepancies regarding the implementations even for the same data formats. Therefore, a large part of molecule editors cannot directly use another editor's knowledge representation. Moreover every molecule editor has its own private mechanism consisting of a set of operations to implement special functionality. Generally, operations of molecule editors can be of two types: manipulating operations and viewing operations. Manipulating operations are a set of commands toward molecule objects, including selection, adding, deletion and modification of properties. These commands are essential for any molecule editor. Usually manipulating operations are semantically equivalent among all molecule editors, no matter how they are implemented. Analogous to the manipulating operations viewing operations are a set of commands to change observation as changing from 2D to 3D, highlighting parts of structure or moving objects in editor's view pane. Those vivid characteristics of different molecule editors always are reflected in this set of operations. According to the varying preferences of biologist and chemists those settings will differ in a collaborative environment. It is of high importance that changing the format of those viewing information does not also modify the view. To achieve this, our in recent work, we do not consider transmission of viewing operations. Our present work discusses how to handle molecule object manipulation information in a heterogeneous environment consistently.

Consistency has always been a very important issue in real time collaborative groupware which allows many users to view and edit the same graphical contents at the same time [21]. This fact is of high

importance in molecule editors, since objects in biology and chemistry usually have more complex structures and as such more operational targets, like atoms, electrons or bonds. So another question we should consider is how to coordinate two or more heterogeneous molecule editors regarding version and maintain the consistency of every client.

The group interface also is an important challenge. Usually there is no "multiple-users" presupposition in a single user molecule editor before we share it. So we need consider how to arrange the collaboration lifecycle, which means starting, joining, ending a collaboration process and coordinating users with heterogeneous editors. Furthermore the organization mode of the collaborating biologist and chemists also needs be taken into account. This includes biology and chemistry computation features and the work habits of biologist and chemists.

4. Implementation

4.1. Collaboration Framework

In general, there are two kinds of collaboration structures we can choose from. Firstly the centralized mode, where all of the shared data is maintained and processed at a single location. Secondly replicated mode, where each client takes charge of maintaining and processing the shared data [22].

Since one copy of the data should be maintained, the centralized architecture has no problem with data consistency [23], but purely centralized systems impose strict What You See Is What I See (WYSIWIS), where the participants see exactly the same view of the shared application at the same time [24]. This means that it is impossible to implement communication between heterogeneous editors, because the shared data cannot be understood by some editors directly and it needs to be converted to a special format which the editor can parse and display. It is better to deal with this operation in replication which is located in the client to avoid unnecessary complexity in the server. If the required conversion is done on the server, a special component needs to be developed to store information on the format capabilities of all involved client-editors. This component then would call different subroutines to convert the source file to corresponding target files. The complexity to distinguish processes increases with the number of users. Furthermore, although the editors we have chosen usually have similar functionality, they all implement large parts of basic operations, like selection, deletion and addition of elements. However actually they all have their own mechanisms to implement these operations, which need to be processed in the client. So it is necessary to adopt parts of the replicated architecture mechanism to make up for a deficiency of the centralized architecture. This can provide some replications for local client to process.

As shown in figure 3, we combine these two structures. The center part only takes charge of transmitting messages to clients, managing users and sessions and providing concurrency control. It is not necessary for the server to know which kind of molecule editors the client integrates and which kind of format the client needs. Comparatively each client takes charge of integrating the molecule editor, converting file formats and some other operations which are not related to the network. Adopting the mixed structure allows us to make the business logic independent of the network transmission.

4.2. Content Coordination of Heterogeneous Editors

We further apply the adapter pattern from computer architectures which different types of equipment transfer their data through a common bus, while adapters can convert data bilaterally.

Adapter can solve the aforementioned problem of semantics by coordinating diverse editors' contents. For each differing editor, there is an adapter to take charge of converting the proprietary format to a general format – CML (Chemical Markup Language) – which is wildly applied by most molecule

applications. In our prototype, we have the two different editors JChemPaint and ChemAxon MarvinSketch.

JCP Editor: JChemPaint (or JCP for short here) is the editor and viewer for 2D molecular structures. The JCP Editor allows the user to draw molecular structures and to import and export structured data in plain-text formats (SMILES [25], MDL[26], and CML).

MarvinSketch: Chemaxon MarvinSketch is an advanced molecule editor for drawing molecular structures. Supported file types include MDL, CML, PDB [27] etc.

Although JCP and MarvinSketch both support the format of CML, which we use as the transmitting, in fact the ontology definitions of their CML-implementations are different. In figure 1 and figure 2, these two CML files represent the same substance, benzene ring. We can see differences ranging from name-space to element definition. Generally speaking they cannot understand each other. In our prototype we define the Marvin Sketch CML format file which can only be analyzed by Marvin Sketch as source file and the general CML format file as result file. Then, we construct a transformation expressed through a stylesheet by using XSLT (eXtensible Stylesheet Language Transforming) [28] and with the stylesheet, the source file can be converted into the resulting file. These operations are organized in a module and processed in the adapter of Marvin Sketch. For other molecule editors, we can develop the special converting module respectively according to the features of the editor.

4.3. Unified User Management

In generally, real-time collaborative editing systems allow a group of users to view and edit the same text, graphics or other multimedia documents, which are molecule objects in this case at the same time from geographically dispersed sites. Therefore, for implementing the collaboration of heterogeneous molecule editors, in particular, for organizing users whose editors have no cooperative functionality. Our prototype provides a unified user management, which adds the conception of multi-user for sharing heterogeneous single-user editors. This allows users to have their own identities during the collaboration processing which is managed in a centralized fashion no matter which kind editor the user has. It also sets up a pattern of running co-work between users, which we call session. This is the unit of collaboration used to organize the collaboration group. Before starting work, users need to join or create a new session. The information regarding a session also is managed by the server. We can see from figure 3 that users A, B, C belong to session one and at the same time, user C also can join in session two to work with user D. The information that user C takes part in every session is easily injected by tagging all messages with user's C identity.

In the transmission of the message in figure 3, messages are transmitted by groups, so messages that belong to different sessions will not be disturbed by each other. The settings of a session enable users launch multiple collaborations with different partners simultaneously. We can also see that user C works in two sessions with different editors.

Fig. 1. Benzene ring represented by JCP

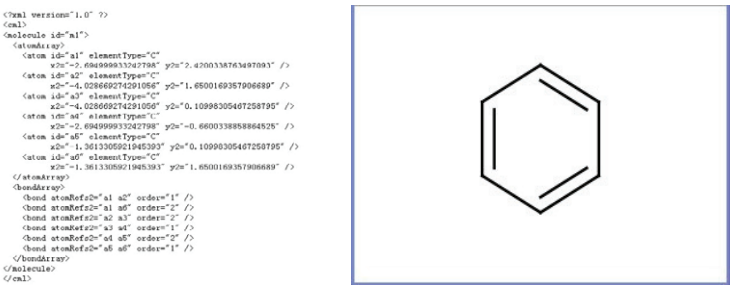

Fig. 2. Benzene ring represented by Marvin Sketch

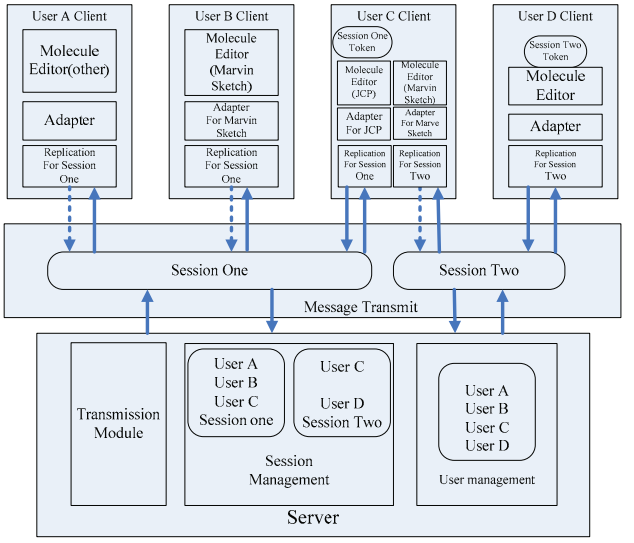

Fig .3. The System Architecture

4.4. Concurrency Control

Furthermore, we design a concurrency control mechanism[29] for both service and client side to provide users with a concurrent method to access the shared data object, like atoms, bonds and other attributes. It also provides the management of version controlling which can always send users the valid and latest information. A modified token-ring algorithm is used to resolve concurrency conflicts.

In collaboration work within one session, only one user can acquire a token. Combining with replication settings located at the client side as a file (different with replication, editors have their own buffer to store the data), when a user with a token approves the modification of a data object, these modifications will be transmitted from the buffer of the editor to replication. So there is a boolean “and” relationship between token and user’s replication, if and only if one user has a token and the replication is changed, the copy of the replication could be transmitted to the server, and then the server will broadcast this copy to other users’ replications that belong to one session. After this, the modification will be reflected to other users’ editors by these editors’ own parse function or adapter. It does not mean only this user can modify the data object, other users also can operate their editors, but their operation only affects the local replication and it will not work for other users.

In figure 3, the full line means that the transmission is effective and the broken line represents that the modification will only be saved in a local replication instead of transmitting it to the server. With the token of session one, only user C can change the shared data and the result would be reflected to clients of user A and B. But in session two, operations of user C cannot affect user D's client who has the token of session two.

With this mechanism, the concurrency conflict has been resolved, and it also avoids the strict WYSIWIS and makes the collaboration pattern more flexible.

5. Evaluation

This section illustrates a scenario of using MarvinSketch (in Figure 4) and JCP (in Figure 5) for collaboration. These two molecule editor are embedded into eclipse platform [30] without any modifications. The main view in Figure 4 and Figure 5 is the workplace for user. In this scene, two users were using their own molecule editor to draw molecular structures. First, they should have registered their identity in server, and then they can use this identity to join in the collaboration work. Second, they should do some preparation work, for example, they need create or find the collaboration session, and upload the existing molecular structure or start working with blank canvas. During the process, although two users had different toolkit provided by different molecule editor, the operations would be transmitted to each other and understood by their editor's own parse mechanism. Definitely, there are some special feature would not be exploited in the collaborative procession because functional differences between those editors. Table 1 show a comparison of operations implemented by and Marvin Sketch (In the line of "JCP" and "Marvin Sketch"). Furthermore, it also shows which operations can be synchronized by our platform and which cannot (In the line of "Synchronization").

6. Conclusion

In the work present we have introduced a framework for a collaboration architecture including unified user management and concurrency control. These concepts have been developed to implement a prototype which has shared two existing molecule editors and made them work together. Our work has demonstrated that by using the transparence approach to share different heterogeneous single-user molecule editors and making them cooperate to each other is a simple but effective way for developing molecule collaboration tools. The presented framework could provide many molecule editors with collaborative functionality. This method is not only useful for biology and chemistry but also beneficial to other domains of science, for example, mathematics, architecture and cybernetics.

We will mainly focus on the research of semantic conversions between different molecule editors' formats in a next step. On this basis, some special format conversion module will be developed for integrating other powerful molecule editors. Finally we also need to do further research regarding the collaboration mode to improve the efficiency of teamwork.

Table 1. Comparison of operations implemented by Marvin Sketch and JCP

	Create New Molecule	***Open Existing*** Molecule	Save Molecule	2D Structure Display	***2D Structure*** Drawing	3D Structure Display	***3D Structure*** Drawing
JChemPaint	Yes	Yes	Yes	Yes	Yes	Yes	No
Marvin Sketch	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Synchronization	Yes	Yes	Yes	Yes	Yes	No	No

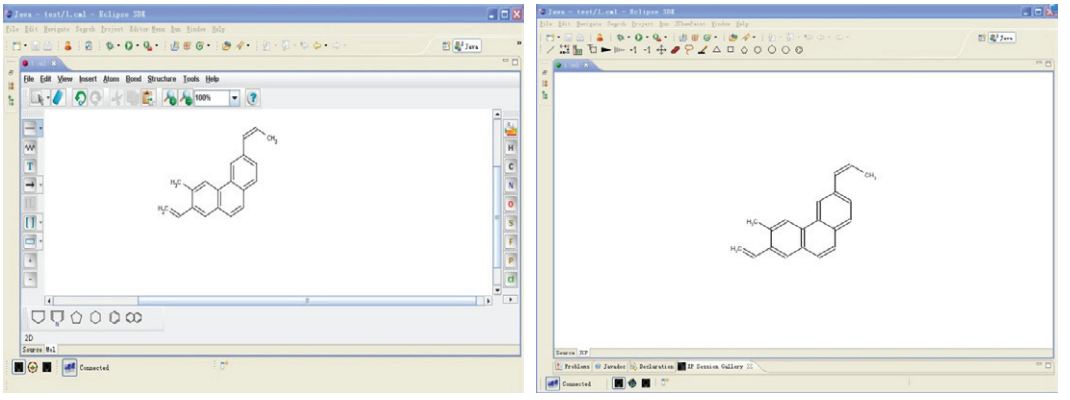

Fig. 4. (a) Marvin Sketch; (b) JCP

Acknowledgements

This work is supported by Natural Science Foundation of P.R.China (90912003). We thank the anonymous reviewers for their comments and suggestions.

References

[1] Ian Foster, Yong. Zhao, Ioan Raicu, and Shiyong Lu. “Cloud Computing and Grid Computing 360-Degree Compared,” Grid Computing Environments Workshop, 2008. GCE '08 (2008), p. 1-10.

[2] Pierfrancesco Bellini, Paolo Nesi, and Marius B Spinu. “Cooperative visual manipulation of music notation,” ACM Transactions on Computer-Human Interaction, 9(3)(2002) p. 194-237.

[3] Du Li and Rui Li, “Preserving Operation Effects Relation in Group Editors,” Proceedings of the 2002 ACM conference on Computer supported cooperative work, November(2002) p. 457-466.

[4] John McMurry. “Organic Chemistry (3rd ed.),” Belmont: Wadsworth

[5] Frank Albert Cotton, Geoffrey Wilkinson, Carlos A Murillo, Manfred Bochmann. “Advanced Inorganic Chemistry (6th ed.),” New York: Wiley-Interscience(1999)

[6] Jonathan Grudin. “Eight challenges for groupware developers,” Communications of the ACM, 37(1) (1994), p. 92–105.

[7] Ansgar R. S. Gerlicher. “A Framework for Real-Time Collaborative Engineering in the Automotive Industries,” Lecture Notes in Computer Science, Springer Berlin / Heidelberg(2006), p. 164 – 173

[8] Du Li, Rui Li. “Transparent Sharing and Interoperation of Heterogeneous Single User Applications,” Proceedings of the 2002, ACM conference on Computer supported cooperative work, (2002),

[9] James Begole, Mary Beth Rosson, and Clifford A. Shaffer. “Flexible Collaboration Transparency: Supporting Worker Independence in Replicated Application-Sharing Systems,” ACM Transactions on Computer-Human Interaction, Vol. 6, No. 2, (1999), p. 95–132.

[10] Jonathan Grudin. “Why CSCW applications fail: Problems in the design and evaluation of organizational interfaces,” In Proceedings of ACM Conference on Computer Supported Cooperative Work (CSCW’88) (1988), p. 85–93

[11] Ronald M. Baecker. “Readings in Groupware and Computer-Supported Cooperative Work,” Morgan Kaufmann Publishers Inc, San Francisco, CA, USA (1992), p. 882

[12] Maxwell D. Cummings, Alan C. Gibbs, Renee L. DesJarlais. “Processing of Small Molecule Databases for Automated Docking,” Medicinal Chemistry 3(1) (2007), p. 107-113.

[13] Ulrich Rester. “From virtuality to reality - Virtual screening in lead discovery and lead optimization: A medicinal chemistry perspective. Current Opinion in Drug Discovery and Development (4) (2008),” p. 559–568.

[14] Stefan Krause, Egon Willighagen and Christoph Steinbeck. “JChemPaint - Using the Collaborative Forces of the Internet to Develop a Free Editor for 2D Chemical Structures,” Molecules (2000), p. 93-98.

[15] ChemAxon. http://www.chemaxon.com/

[16] Peter W. Kenny, Jens Sadowski. “Structure modification in chemical databases,” Chemoinformatics in drug discovery (editor—Oprea, T.I.) (2005), p. 271-285

[17] Michael J. Knister and Atul Prakash. “DistEdit: A distributed toolkit for supporting multiple group editors,” In Proceedings of ACM CSCW’90 Conference on Computer Supported CooperativeWork(1990), p. 343– 355..

[18] Kevin Palfreyman, Tom Rodden, and Jonathan Trevor. “PSI: A platform for shared interaction,” In ECSCW 99: Proceedings of the Sixth European Conference of Computer Supported Cooperative Work (1999), p. 351–370.

[19] Clarence A. Ellis, Simon J. Gibbs, Gail Rein. “Groupware Some Issues and Experiences,” Communications of the ACM, Volume 34 , Issue 1 (1991), p. 39 - 58

[20] Chemical Markup Language. http://cml.sourceforge.net/

[21] Chengzheng Sun and David Chen. “Consistency Maintenance in Real-Time Collaborative Graphics Editing Systems,” ACM Transactions on Computer-Human Interaction, Vol. 9, No. 1 (2002), p. 1–41

[22] George Coulouris, Jean Dollimore, and Tim Kindberg. “Distributed Systems: Concepts and Design. Reading,” MA: Addison-Wesley(1994)

[23] James Begole, Randall B. Smith, Craig A. Struble, and Clifford A. Shaffer. “Resource Sharing for Replicated Synchronous Groupware,” IEEE/ACM Transactions on Networking, vol. 9, No. 6(2001), p. 833-843.

[24] Mark Jeffrey Stefik, Daniel G Bobrow, Greg Stanley Foster, Stan Lanning, and Deborah G Tatar. “WYSIWIS revised: Early experiences with multiuser interfaces”. ACM Trans. Office Inform. Syst, vol. 5, No. 2(1987), p. 147–167.

[25] SMILES. http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html

[26] Mol. http://www.symyx.com/solutions/white_papers/ctfile_formats.jsp

[27] Helen M. Berman. “The Protein Data Bank: a historical perspective,”Acta Crystallographica Section A: Foundations of Crystallography (2008)A64 (1): p. 88–95

[28] XSLT. http://xmlxslt.sourceforge.net/

[29] Jonathan Munson and Prasun Dewan. “A Concurrency Control Framework for Collaborative Systems,” In Proc of ACM Conf on Computer Supported Cooperative Work(1996) ,p. 278-287.

[30] Eclipse. http://www.eclipse.org/.